Evaluation of Classification Results for Convolutional Neural Network Architectures on the MNIST and CIFAR-10 Datasets Using TensorFlow

[1]Denny Nur Ramadhan, [2]Prof. Dr. Eng. Cahya Rahmad, ST., M.Kom., [3]Yan Watequlis Syaifudin, ST., M.MT., Ph.D.

[1]Master Student, [2]Professor, [3]Professor
[1]Applied Master of Information Technology Engineering,
[1]State Polytechnic of Malang, Malang, Indonesia
[1]denny.dand@outlook.com, [2]cahya.rahmad@polinema.ac.id, [3]qulis@polinema.ac.id

Abstract— This study evaluates 40 convolutional neural network (CNN) architectures from TensorFlow on MNIST and CIFAR-10 datasets representing different visual complexities. Under unified experimental conditions, models from ConvNeXt, EfficientNet, ResNet, DenseNet, MobileNet, and other families were assessed for accuracy, loss, weighted metrics, training time, and memory usage. Results show distinct architectural effectiveness: on MNIST, most models achieved near-saturation performance (24 exceeding 98% accuracy) with minimal dispersion, indicating lightweight architectures suffice for simple tasks. On CIFAR-10, performance varied substantially (67.21–96.28% accuracy), with modern, capacity-rich architectures like ConvNeXtXLarge excelling but requiring greater computational resources. Comparative analysis reveals dataset-dependent optimality—EfficientNetV2 dominates MNIST while ConvNeXt excels on CIFAR-10—and demonstrates that architectural complexity yields meaningful returns only on visually complex datasets. These findings provide a practical framework for model selection based on task complexity and resource constraints.

Index Terms—Convolutional Neural Networks, Image Classification, MNIST, CIFAR-10, TensorFlow, Deep Learning Benchmark.

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have become a cornerstone of modern image classification due to their ability to automatically learn hierarchical feature representations from visual data. Over the past decade, CNN-based approaches have demonstrated remarkable success across a wide range of computer vision tasks, including handwritten digit recognition, object classification, and facial analysis. These advances have been driven not only by increased computational power but also by the rapid development of diverse CNN architectures with varying design philosophies.

As deep learning research has progressed, numerous CNN architectures have been proposed, each introducing different trade-offs in terms of classification accuracy, parameter efficiency, training stability, and computational cost. In practical applications, selecting an appropriate CNN architecture is rarely straightforward, as models that achieve high accuracy often require substantial computational resources, while lightweight architectures may sacrifice representational capacity. This challenge is further amplified by the availability of pretrained CNN models provided by modern deep learning frameworks, which offer a wide range of architectural options for transfer learning.

TensorFlow has emerged as one of the most widely adopted deep learning frameworks [1], providing an extensive collection of pretrained CNN architectures through its Keras Applications module. While these models facilitate rapid experimentation and deployment, their sheer number and architectural diversity introduce uncertainty regarding how different CNN designs behave when applied to datasets with varying levels of visual complexity. Although many studies focus on proposing novel architectures or improving state-of-the-art performance, relatively few works conduct a systematic empirical evaluation of multiple CNN architectures under a unified experimental configuration, particularly within a single framework such as TensorFlow.

To address this gap, this study conducts a comprehensive evaluation of classification results obtained from a broad range of CNN architectures implemented using TensorFlow. The evaluation is performed on two widely used benchmark datasets—MNIST and CIFAR-10—which represent image classification tasks with substantially different levels of visual complexity. By applying identical preprocessing pipelines, training configurations, and evaluation metrics across all models, this study aims to provide a fair and consistent comparison of CNN behavior. Rather than proposing new architectures, the focus of this work is to analyze classification outcomes, generalization behavior, and computational trade-offs, thereby offering practical insights into model selection for image classification tasks of varying complexity.

II. RELATED WORK

Numerous studies have explored the performance of different CNN architectures on standard image classification datasets. Notably, the original research introducing architectures such as AlexNet [2], VGG [3], Inception [4], ResNet [5], DenseNet [6], and EfficientNet [7] laid the foundation for rapid progress in the field. Each of these models introduced architectural innovations to improve accuracy, reduce parameters, or enhance training stability.

Chen et al. [8] presented an extensive review of image classification algorithms based on CNNs. Their work compared the performance of various CNN architectures, including VGG, ResNet, Inception, DenseNet, MobileNet, and EfficientNet, across multiple datasets. However, their analysis was conducted as a literature survey, aggregating results from different studies without direct experimentation under a unified environment. Consequently, factors such as training parameters, preprocessing techniques, and dataset splits were inconsistent across models, limiting the comparability of the findings.

In another work, Mostafa et al. [9] benchmarked several CNN models on a facial expression recognition task using cloud-based environments. While the study provides valuable insights into model accuracy and training efficiency, its scope was restricted to models available in Keras at the time, and focused solely on facial expression datasets. Additionally, modern architectures such as ConvNeXt and NASNet were not evaluated. The study also involved training in multiple sessions, which, while practical in shared cloud environments such as Google Colab, may lead to minor variations in training dynamics due to stochastic optimization processes and session-level execution differences, rather than numerical precision issues.

In contrast, this study conducts a unified and consistent evaluation of classification results across multiple CNN architectures implemented using TensorFlow. All models are trained under identical experimental settings on two widely used image classification datasets, MNIST and CIFAR-10, which represent different levels of visual complexity. A consistent training configuration, preprocessing pipeline, batch size, and early stopping criteria are applied to ensure fair comparison. By focusing on classification outcomes rather than proposing new architectures, this study provides an empirical analysis of CNN behavior across datasets with varying complexity within a unified TensorFlow-based environment.

III. METHODOLOGY

3.1. Dataset Preparation

The experiments in this study were conducted using two widely adopted image classification datasets, MNIST [10] and CIFAR-10 [11], which are commonly used as standard benchmarks for evaluating convolutional neural network (CNN) architectures. The CIFAR-10 dataset consists of 60,000 color images with a resolution of 32×32 pixels, evenly distributed across 10 object classes, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is pre-divided into 50,000 training images and 10,000 test images. To support model evaluation and training control, the training set was further split into 40,000 images for training and 10,000 images for validation.

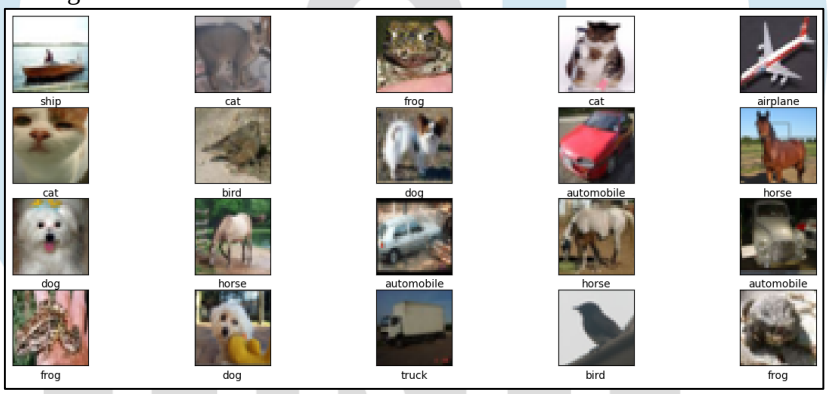

Figure 1. Example CIFAR-10 images representing different classes

The MNIST dataset contains grayscale images of handwritten digits ranging from 0 to 9, with an original resolution of 28×28 pixels. Similar to CIFAR-10, the dataset provides predefined training and test splits, which are widely used in the literature for benchmarking image classification models on simpler visual patterns.

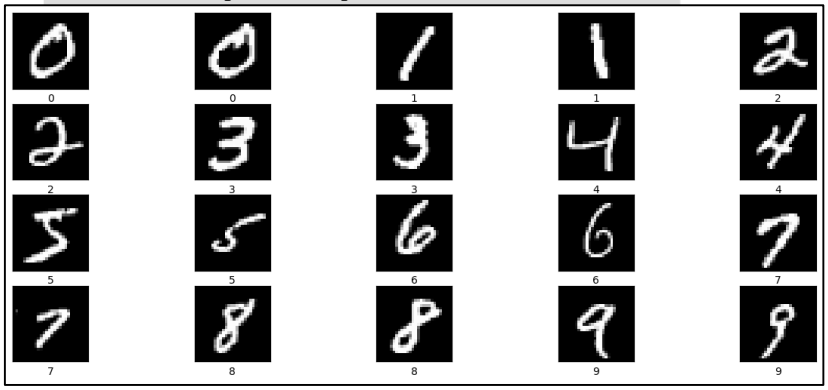

Figure 2. Example CIFAR-10 images representing different classes

3.2. Model Coverage and Transfer Learning Setup

This study evaluates a broad range of convolutional neural network (CNN) architectures available in TensorFlow 2.17.0 to ensure comprehensive coverage of commonly used architectural families within the framework [12]. A total of 40 models were included, representing a diverse range of architectural families such as ConvNeXt [13], DenseNet [6], EfficientNet [7] EfficientNetV2 [14], MobileNet [15], MobileNetV2 [16], MobileNetV3 [17], NASNet [18], ResNet [5], ResNetV2 [19], VGG [3], InceptionV3 [19], InceptionResNetV2 [20] and Xception [21].

Rather than selectively benchmarking only a small subset of popular models, this study adopts an inclusive evaluation strategy to observe how different architectural designs behave under identical training conditions. All models are trained using the same experimental configuration, including preprocessing pipeline, optimizer settings, batch size, and early stopping criteria. This unified setup enables a fair comparison of classification results while minimizing confounding factors introduced by inconsistent training environments.

Table 1. List of CNN Architectures Available in Tensorflow

No	Model	Parameters	No	Model	Parameters
1	ConvNeXtBase	87,831,434	21	EfficientNetV2L	118,077,354
2	ConvNeXtLarge	196,626,378	22	EfficientNetV2M	53,480,894
3	ConvNeXtSmall	49,654,122	23	EfficientNetV2S	20,661,866
4	ConvNeXtTiny	28,019,562	24	InceptionResNetV2	54,732,778
5	ConvNeXtXLarge	348,675,082	25	InceptionV3	22,329,898
6	DenseNet121	7,302,474	26	MobileNet	3,493,834
7	DenseNet169	13,071,690	27	MobileNetV2	2,588,490
8	DenseNet201	18,816,330	28	MobileNetV3Large	3,244,938
9	EfficientNetB0	4,380,077	29	MobileNetV3Small	1,089,402
10	EfficientNetB1	6,905,745	30	NASNetLarge	85,951,836
11	EfficientNetB2	8,131,843	31	NASNetMobile	4,542,878
12	EfficientNetB3	11,179,577	32	ResNet50	24,114,826
13	EfficientNetB4	18,135,401	33	ResNet50V2	24,091,914
14	EfficientNetB5	29,040,641	34	ResNet101	43,185,290
15	EfficientNetB6	41,552,793	35	ResNet101V2	43,153,674
16	EfficientNetB7	64,755,873	36	ResNet152	58,898,058
17	EfficientNetV2B0	6,249,818	37	ResNet152V2	58,858,762
18	EfficientNetV2B1	6,905,745	38	VGG16	14,848,586
19	EfficientNetV2B2	9,132,648	39	VGG19	20,158,282
20	EfficientNetV2B3	13,326,664	40	Xception	21,388,594

3.3. Experimental Setup

The experiments were conducted using two widely adopted image classification datasets, MNIST and CIFAR-10, which represent datasets with different levels of visual complexity. MNIST consists of grayscale images of handwritten digits, while CIFAR-10 comprises color images from ten object categories. To ensure compatibility with the input requirements of the pretrained convolutional neural network architectures, both datasets were resized to 224 × 224 pixels, as most of the evaluated models were originally trained on ImageNet using this input resolution. The MNIST dataset contains a total of 60,000 images, which were divided into 48,996 training samples, 10,494 validation samples, and 10,510 test samples. The CIFAR-10 dataset consists of 60,000 color images distributed across ten classes and was split into 40,000 training images, 10,000 validation images, and 10,000 test images. For both datasets, the subsets were organized using a directory structure compatible with standard image classification pipelines.

During training, data were fed into the models using a batch size of 32, with categorical labels. Input preprocessing was performed using each model's corresponding preprocessing function provided by the TensorFlow library, ensuring consistency with the original normalization schemes applied during ImageNet pretraining. All models were trained using the Adam optimizer with categorical cross-entropy loss. A maximum of 100 epochs was specified for each experiment, with early stopping enabled based on validation loss using a patience value of 10 epochs, in order to prevent overfitting and reduce unnecessary training time. All experiments were executed on a single local computing environment to maintain consistency in hardware and software conditions throughout the study. The system was equipped with an AMD Ryzen 5 7600X CPU and an NVIDIA GeForce RTX 3070 GPU. This setup allowed all models to be trained under identical execution conditions, minimizing variability introduced by changes in hardware allocation or runtime environments.

3.4. Evaluation Protocol

To evaluate the classification results of each Convolutional Neural Network (CNN) architecture, several standard evaluation metrics were employed. These metrics were selected to provide both overall and class-level insights into model behavior on the MNIST and CIFAR-10 datasets. Classification accuracy was used as the primary metric to measure overall performance. Accuracy is defined as the ratio of correctly predicted samples to the total number of samples in the test set, reflecting the model's ability to correctly classify instances across all classes. In addition to accuracy, classification loss was measured using the categorical cross-entropy (CCE) loss function, which is commonly applied in multi-class classification tasks. This loss function evaluates the discrepancy between predicted class probability distributions and the one-hot encoded ground truth labels by computing the negative log-likelihood of the correct class. Categorical cross-entropy is widely adopted due to its effectiveness in probabilistic neural network training.

To provide a more detailed analysis beyond overall accuracy, this study also reports weighted precision, weighted recall, and weighted F1-score. Precision represents the proportion of correctly predicted positive samples relative to all predicted positives, while recall measures the proportion of correctly predicted positive samples relative to all actual positives in the ground truth. The F1-score, defined as the harmonic mean of precision and recall, provides a balanced measure that accounts for both false positives and false negatives. These metrics were computed by aggregating predictions across the entire test set and constructing a confusion matrix for each model. The confusion matrix enables an examination of class-wise prediction behaviour and common misclassification patterns. Weighted averaging was adopted to summarize precision, recall, and F1-score by assigning weights proportional to the number of samples in each class. This approach reflects the overall classification performance more realistically when class frequencies may differ across datasets or experimental splits. In addition, confusion matrices were analysed for selected models to further illustrate prediction behaviour and identify systematic classification errors, particularly in cases where visually similar classes may lead to higher confusion.

.

IV. Results and Discussion

4.1. *Result on the CIFAR-10 Dataset*

The CIFAR-10 results (Table 3) show substantial architectural differentiation, with accuracy ranging from 67.21% (ResNet152) to 96.28% (ConvNeXtXLarge). Modern architectures—particularly the ConvNeXt family—dominated the rankings, with ConvNeXtXLarge, Large, and Base achieving 96.28%, 95.83%, and 95.76% respectively. In contrast, lightweight models like MobileNetV2 (84.88%) and ResNet50 (73.36%) exhibited notable limitations. The confusion matrix of ConvNeXtXLarge (Figure 3) reveals its strong diagonal concentration with minimal cross-category confusion, primarily between semantically related pairs like cat/dog and automobile/truck.

Table 2. Classification Results of CNN Architectures on the CIFAR-10 Dataset

Model	Accuracy	Loss	Training Time (m)	Memory Usage (MB)	Weighted Precision	Weighted Recall	Weighted F1
ConvNeXtBase	0.9576	0.1279	29.08	2028.16	0.96	0.96	0.96
ConvNeXtLarge	0.9583	0.1273	75.33	1465.81	0.96	0.96	0.96
ConvNeXtSmall	0.9345	0.1985	22.95	1816.39	0.94	0.93	0.93
ConvNeXtTiny	0.9177	0.241	14.85	1352.43	0.92	0.92	0.92
ConvNeXtXLarge	0.9628	0.107	88.2	3789.46	0.96	0.96	0.96
DenseNet121	0.8894	0.3231	11.54	1413.82	0.89	0.89	0.89
DenseNet169	0.8822	0.3393	13.33	883.79	0.89	0.88	0.88
DenseNet201	0.9034	0.2799	16.42	1891.8	0.9	0.9	0.9
EfficientNetB0	0.9087	0.2897	6.9	1303.47	0.91	0.91	0.91
EfficientNetB1	0.9165	0.2469	8.69	1666.8	0.92	0.92	0.92
EfficientNetB2	0.9212	0.2388	8.2	1579.17	0.92	0.92	0.92
EfficientNetB3	0.9255	0.2291	11.45	1808.03	0.93	0.93	0.93
EfficientNetB4	0.9242	0.2303	14.12	2028.91	0.92	0.92	0.92
EfficientNetB5	0.9179	0.2578	18.58	2212.76	0.92	0.92	0.92
EfficientNetB6	0.9143	0.2552	25.09	2493.5	0.91	0.91	0.91
EfficientNetB7	0.9175	0.2481	32.67	3182.58	0.92	0.92	0.92
EfficientNetV2B0	0.9125	0.2545	5.82	1393.77	0.91	0.91	0.91
EfficientNetV2B1	0.9269	0.2193	7.17	1471.99	0.93	0.93	0.93
EfficientNetV2B2	0.928	0.2286	8.98	1695.58	0.93	0.93	0.93
EfficientNetV2B3	0.9355	0.2026	10.85	1785.55	0.94	0.94	0.94
EfficientNetV2L	0.9389	0.1962	39.61	4001.57	0.94	0.94	0.94
EfficientNetV2M	0.9268	0.2252	25.92	2858.78	0.93	0.93	0.93
EfficientNetV2S	0.9248	0.2267	14.9	1950.4	0.93	0.92	0.92
InceptionResNetV2	0.9059	0.2884	23.71	2106.88	0.91	0.91	0.91
InceptionV3	0.8664	0.3967	8.94	1257.7	0.87	0.87	0.87
MobileNet	0.8741	0.367	4.03	914.47	0.88	0.87	0.87
MobileNetV2	0.8488	0.4506	5.05	1011.58	0.85	0.85	0.85
MobileNetV3Large	0.8982	0.301	3.73	1159.81	0.9	0.9	0.9
MobileNetV3Small	0.8862	0.3345	2.45	1246.92	0.89	0.89	0.89
NASNetLarge	0.8932	0.3237	39.6	2730.03	0.89	0.89	0.89
NASNetMobile	0.8594	0.4161	8.77	1808.51	0.86	0.86	0.86
ResNet101	0.6821	0.9272	53.56	2171.94	0.69	0.68	0.68
ResNet101V2	0.8954	0.3108	14.84	1765.67	0.9	0.9	0.89
ResNet152	0.6721	0.9389	98.59	2563.75	0.67	0.67	0.67
ResNet152V2	0.8857	0.3395	21.81	1987.68	0.89	0.89	0.89
ResNet50	0.7336	0.766	18.97	1306.91	0.74	0.73	0.73
ResNet50V2	0.8632	0.3821	8.44	1194.11	0.87	0.86	0.86
VGG16	0.8548	0.4295	15.49	1003.36	0.86	0.85	0.85
VGG19	0.8527	0.4267	20.86	894.39	0.86	0.85	0.85
Xception	0.8665	0.4056	15.31	1194.94	0.87	0.87	0.87

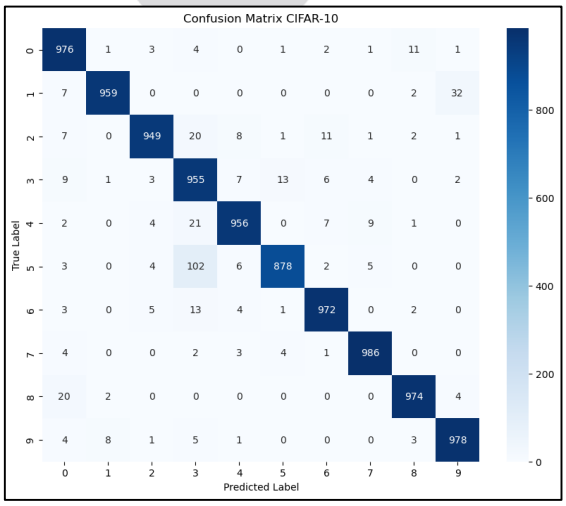

Figure 3. Confusion matrix of ConvNeXtXLarge on the CIFAR-10 test set

Computational requirements varied dramatically: ConvNeXtXLarge required 88.2 minutes and 3789 MB, while MobileNetV3Small completed training in just 2.45 minutes with 1247 MB—a 36× speed advantage but with a 7.46 percentage point accuracy trade-off. EfficientNetV2B3 offered a compelling middle ground with 93.55% accuracy in only 10.85 minutes

4.2. Result on the MNIST Dataset

This section presents the evaluation results on the MNIST dataset, which represents a simple image classification task with grayscale handwritten digits and limited intra-class variation. Under identical experimental conditions, most CNN architectures achieved near-saturation performance, demonstrating that MNIST's low visual complexity allows even lightweight models to approach optimal classification. As shown in Table 3, classification accuracy was consistently high across architectures, with 24 of 40 models exceeding 98% accuracy. The top-performing model, EfficientNetV2B2, achieved 98.65% accuracy with the lowest loss (0.0445), followed closely by EfficientNetV2B3 (98.49%) and EfficientNetV2B1 (98.41%). This performance concentration indicates that MNIST presents minimal discriminative challenge for modern CNNs, regardless of architectural sophistication. The confusion matrix of EfficientNetV2B2 (Figure 4) visually confirms this near-perfect performance, showing strong diagonal concentration with only minor misclassifications—primarily between visually similar digit pairs such as '4'/'9' and '7'/'1'. Weighted precision, recall, and F1-score values were uniformly high across models, with 35 architectures achieving F1-scores of 0.98 or higher, confirming balanced performance across all digit classes

Table 3. Classification Results of CNN Architectures on the MNIST Dataset

Model	Accuracy	Loss	Training Time (m)	Memory Usage (MB)	Weighted Precision	Weighted Recall	Weighted F1
ConvNeXtBase	0.9821	0.0634	72.2	2707.21	0.98	0.98	0.98
ConvNeXtLarge	0.9799	0.0674	103.58	3148.87	0.98	0.98	0.98
ConvNeXtSmall	0.9808	0.0649	47.41	2288.92	0.98	0.98	0.98
ConvNeXtTiny	0.9814	0.0642	28.18	1803.6	0.98	0.98	0.98
ConvNeXtXLarge	0.975	0.0832	162.48	4171.13	0.98	0.97	0.98
DenseNet121	0.976	0.0959	20.95	1635.69	0.98	0.98	0.98
DenseNet169	0.978	0.075	20.75	1975.06	0.98	0.98	0.98
DenseNet201	0.9799	0.0691	25.31	2071.29	0.98	0.98	0.98
EfficientNetB0	0.9811	0.0587	9.77	1210.5	0.98	0.98	0.98
EfficientNetB1	0.9775	0.0725	17.45	1726.88	0.98	0.98	0.98
EfficientNetB2	0.9804	0.0674	23.1	1855.27	0.98	0.98	0.98
EfficientNetB3	0.9812	0.0634	67.23	2275.85	0.98	0.98	0.98
EfficientNetB4	0.9816	0.06	41.56	2567.82	0.98	0.98	0.98
EfficientNetB5	0.9789	0.0649	39.61	2573.93	0.98	0.98	0.98
EfficientNetB6	0.9788	0.0665	49.54	2780.47	0.98	0.98	0.98
EfficientNetB7	0.9837	0.0629	92.25	3650.66	0.98	0.98	0.98
EfficientNetV2B0	0.9833	0.0525	14.09	1779.56	0.98	0.98	0.98
EfficientNetV2B1	0.9841	0.0526	22.67	1907.38	0.98	0.98	0.98
EfficientNetV2B2	0.9865	0.0445	33.18	2159.14	0.99	0.99	0.99
EfficientNetV2B3	0.9849	0.0506	21.59	2341.34	0.98	0.98	0.98
EfficientNetV2L	0.9798	0.0599	128.79	4864.96	0.98	0.98	0.98
EfficientNetV2M	0.9768	0.0746	102.88	3600.68	0.98	0.98	0.98
EfficientNetV2S	0.982	0.0581	32.68	2348.98	0.98	0.98	0.98
InceptionResNetV2	0.9664	0.1091	36.03	2178.47	0.97	0.97	0.97
InceptionV3	0.9666	0.0992	10.84	1348.12	0.97	0.97	0.97
MobileNet	0.9809	0.0728	6.9	1065.52	0.98	0.98	0.98
MobileNetV2	0.9743	0.0914	6.42	1151.62	0.97	0.97	0.97
MobileNetV3Large	0.9763	0.0847	6.83	1436.26	0.98	0.98	0.98
MobileNetV3Small	0.9782	0.0767	4.32	1282.22	0.98	0.98	0.98
NASNetLarge	0.9626	0.1287	63.43	2682.77	0.96	0.96	0.96
NASNetMobile	0.9601	0.137	13.46	1931.65	0.96	0.96	0.96
ResNet101	0.9747	0.0744	72.15	2424.28	0.97	0.97	0.97
ResNet101V2	0.9826	0.0661	28.34	1743.58	0.98	0.98	0.98
ResNet152	0.9677	0.101	65.63	2434.82	0.97	0.97	0.97
ResNet152V2	0.9768	0.0773	32.59	2144.85	0.98	0.98	0.98
ResNet50	0.9834	0.0536	31.59	1618.82	0.98	0.98	0.98
ResNet50V2	0.9791	0.0783	13.48	1538.55	0.98	0.98	0.98
VGG16	0.975	0.1054	25.62	1099.08	0.98	0.97	0.98
VGG19	0.9775	0.0746	29.71	1232.87	0.98	0.98	0.98
Xception	0.968	0.1165	23.21	1385.18	0.97	0.97	0.97

Despite similar classification outcomes, computational requirements varied substantially. Lightweight architectures demonstrated significant efficiency advantages: MobileNet completed training in only 6.9 minutes with 1065.52 MB memory usage, while MobileNetV3Small required just 4.32 minutes. In contrast, larger models like ConvNeXtXLarge needed 162.48 minutes (23.5× longer) and 4171.13 MB memory, yet delivered only 97.5% accuracy—a minimal 1.15 percentage point improvement over much faster alternatives.

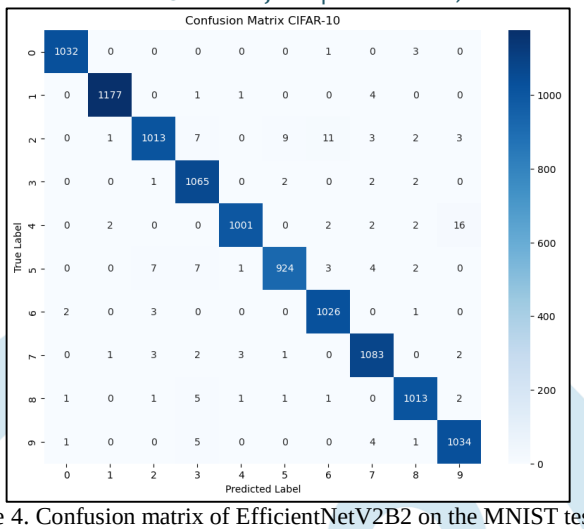

Figure 4. Confusion matrix of EfficientNetV2B2 on the MNIST test set

This pattern reveals clear diminishing returns: increased architectural complexity on MNIST yields negligible accuracy gains despite substantially higher computational costs. The results suggest that for simple, well-separated classification tasks like digit recognition, efficiency-focused architectures provide optimal balance, as representational capacity beyond a basic threshold offers limited practical benefit.

4.3. *Comparative Analysis between CIFAR-10 and MNIST*

The comparative analysis between the experimental results obtained on the CIFAR-10 and MNIST datasets reveals fundamental differences in how convolutional neural network architectures respond to varying levels of visual complexity, providing critical insights into the relationship between dataset characteristics, architectural design, and model performance. This comparison highlights that while modern CNN architectures are universally capable of achieving high performance on simple image classification tasks, their relative efficacy and the necessity of architectural sophistication become substantially more pronounced when confronted with datasets containing greater visual diversity and inter-class ambiguity. The most striking distinction emerges in the distribution of classification accuracy across models: on MNIST, 40 models achieved an average accuracy of 97.98% with a standard deviation of merely 0.009, whereas on CIFAR-10, the average accuracy dropped to 88.54% with a significantly larger standard deviation of 0.078, indicating not only lower overall performance but also greater variability in model capability.

This divergence underscores that the homogeneity and low-dimensional feature space of MNIST allow even lightweight architectures to approach near-optimal performance, while the heterogeneous nature of CIFAR-10 exposes the limitations of simpler models and rewards architectures with greater representational capacity. Further examination of performance trends reveals that architectural families that excelled on one dataset did not necessarily maintain superiority on the other, illustrating the task-dependent nature of model effectiveness. For instance, the EfficientNetV2 series demonstrated exceptional performance on MNIST, occupying the top three positions with accuracies above 98.41%, yet on CIFAR-10, while still competitive, they were surpassed by the ConvNeXt family, particularly the XLarge variant which achieved the highest accuracy of 96.28%. Conversely, some architectures exhibited inconsistent cross-dataset performance; ResNet50, for example, achieved a strong 98.34% accuracy on MNIST but only 73.36% on CIFAR-10 – a performance drop of nearly 25 percentage points that highlights potential limitations in feature extraction robustness when transitioning from simple to complex visual domains.

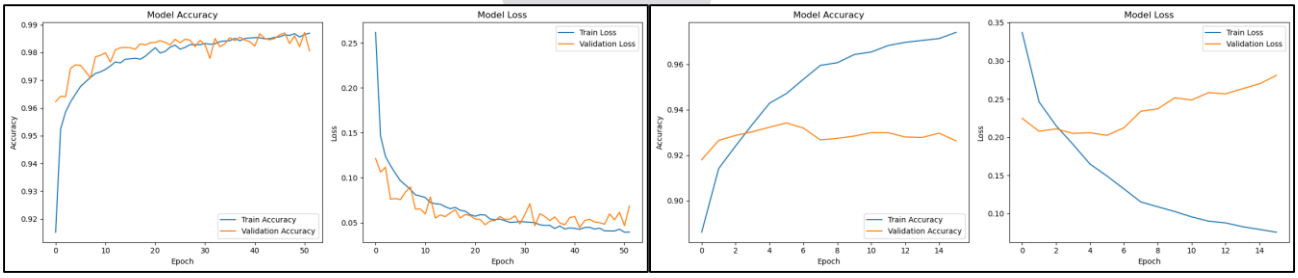

Figure 5. Accuracy and loss curves of EfficientNetV2B2 across MNIST (left) and CIFAR-10 (right) datasets

The training dynamics of EfficientNetV2B2, visualized in Figure 5, provide compelling graphical evidence of this dataset-dependent behavior. On MNIST (Figure 5 - left), the model demonstrates near-perfect generalization, with training and validation accuracy curves rising in tight alignment to approximately 99% and loss values decreasing synchronously to stable minima. In stark contrast, the same architecture on CIFAR-10 (Figure 5 - right) exhibits classic signs of overfitting: a persistent and widening gap emerges between training accuracy (~97%) and validation accuracy (plateauing around 92–93%), while validation loss increases despite decreasing training loss. This visual divergence explains why architectural capacity becomes critical for complex datasets—models must not only learn features but also resist overfitting through better regularization and representational power.

This pattern suggests that architectural innovations optimized for parameter efficiency and gradient flow (as seen in EfficientNetV2) may be sufficient for well-structured, low-variability tasks, but more extensive modeling capacity and hierarchical feature integration (as embodied by ConvNeXt) become increasingly valuable when dealing with real-world visual complexity. The computational resource requirements and efficiency trade-offs also exhibited dataset-dependent characteristics that warrant careful consideration. On MNIST, the relationship between model complexity and accuracy demonstrated clear diminishing returns, with the most computationally intensive model (ConvNeXtXLarge) requiring 5.8 times longer training time

than the fastest model (MobileNet) while delivering only a 1.15 percentage points accuracy improvement. In contrast, on CIFAR-10, the same ConvNeXtXLarge model required 36 times longer training than the fastest model (MobileNetV3Small) but achieved a substantial 7.46 percentage points accuracy advantage, indicating that increased computational investment yields more meaningful performance gains on complex datasets. This differential return on investment fundamentally alters the cost-benefit analysis for model selection: for MNIST-like tasks, efficiency considerations may rightfully dominate, whereas for CIFAR-10-like applications, accepting higher computational costs may be justified by the corresponding improvements in classification quality.

Additionally, the consistency of classification performance across metrics provides further insight into architectural robustness. On MNIST, the alignment between accuracy, weighted F1-score, and loss values was remarkably consistent across nearly all architectures, with 35 of 40 models achieving F1-scores within 0.5% of their accuracy values. On CIFAR-10, however, this alignment showed greater variability, particularly for models with intermediate performance, suggesting that some architectures may achieve comparable accuracy through different mechanisms—some maintaining balanced precision and recall across classes, while others excelling on certain categories at the expense of others. This metric divergence reinforces that dataset complexity not only affects overall performance but also influences the qualitative nature of how models arrive at their predictions.

V. Results and Discussion

This study conducted a comprehensive empirical evaluation of 40 convolutional neural network (CNN) architectures implemented in TensorFlow, examining their classification performance, generalization behavior, and computational efficiency on two benchmark datasets with fundamentally different visual complexities: MNIST and CIFAR-10. Under a unified experimental configuration—encompassing identical preprocessing pipelines, training protocols, and evaluation metrics—the research provides systematic insights into how architectural design choices interact with dataset characteristics to determine model effectiveness.

The results reveal a clear dichotomy in model behavior across datasets. On MNIST, a dataset characterized by low intra-class variation and high inter-class separation, nearly all architectures achieved near-saturation performance, with 24 models exceeding 98% accuracy and the top-performing EfficientNetV2B2 reaching 98.65%. The minimal performance dispersion (standard deviation: 0.009) and consistent alignment between training and validation metrics indicate that lightweight, parameter-efficient architectures are not only sufficient but often optimal for such tasks, as increased model complexity yields diminishing returns with substantially higher computational costs. This suggests that for simple visual recognition problems, considerations of training efficiency and deployment resource constraints should dominate architectural selection.

In contrast, CIFAR-10—with its color images, complex textures, diverse backgrounds, and semantically overlapping categories—exposed significant architectural differentiation. Performance varied widely (accuracy range: 67.21%–96.28%; standard deviation: 0.078), with modern, capacity-rich architectures like ConvNeXtXLarge achieving superior accuracy (96.28%) while simpler models demonstrated notable limitations. The training dynamics further illustrated this divergence: whereas models on MNIST exhibited tight coupling between training and validation curves, CIFAR-10 training revealed characteristic signs of overfitting, including widening accuracy gaps and divergent loss behavior. This indicates that complex visual domains demand architectures with greater representational capacity and intrinsic regularization mechanisms to bridge the generalization gap.

The comparative analysis underscores that optimal CNN selection cannot be determined independently of dataset characteristics. Architectural families that excelled on one dataset did not necessarily maintain superiority on the other: EfficientNetV2 variants dominated MNIST rankings but were surpassed by ConvNeXt models on CIFAR-10, while some architectures like ResNet50 showed dramatic performance drops (25 percentage points) when transitioning from simple to complex visual domains. Furthermore, computational trade-offs exhibited dataset-dependent value propositions: on MNIST, the most expensive model offered minimal accuracy gains (1.15 percentage points) at 5.8× greater training time, whereas on CIFAR-10, the same investment yielded substantial improvements (7.46 percentage points) despite 36× longer training.

These findings offer several practical implications for researchers and practitioners. First, they provide a principled framework for model selection based on task complexity: for MNIST-like problems with clear visual separability, efficiency-focused architectures (MobileNet, EfficientNetB0) provide optimal balance; for CIFAR-10-like challenges requiring fine-grained discrimination, capacity-rich models (ConvNeXt, EfficientNetV2L) justify their computational overhead. Second, the results caution against extrapolating architectural performance across dissimilar domains, highlighting the need for task-specific benchmarking rather than reliance on general leaderboards. Finally, the study demonstrates the value of holistic evaluation encompassing not only accuracy but also training dynamics, resource consumption, and generalization behavior.

Future work could extend this analysis along several dimensions: evaluating architectures on more diverse and challenging datasets (e.g., ImageNet, domain-specific medical or satellite imagery), examining the impact of advanced training techniques (data augmentation, regularization, optimization strategies), and exploring adaptive architectures that dynamically adjust their complexity based on input characteristics. Additionally, investigating the correlation between architectural properties (depth, width, connectivity patterns) and specific dataset attributes could lead to more principled model-design guidelines.

In summary, this research contributes a systematic, empirically grounded comparison of CNN architectures within a consistent TensorFlow environment, providing actionable insights for model selection across varying visual complexity levels. By elucidating the interplay between architectural design, dataset characteristics, and performance outcomes, the study advances practical understanding of deep learning model evaluation and offers a validated reference for future research and application development in image classification.

References

[1] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” May 2016.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Adv Neural Inf Process Syst*, vol. 25, 2012.
[3] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Apr. 2015.
[4] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[6] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," Jan. 2018.
[7] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020.
[8] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," Nov. 01, 2021, *MDPI*. doi: 10.3390/rs13224712.
[9] A. Hesham Mostafa, H. Abdel-galil El-sayed, and M. Abdel-fattah Belal, "BENCHMARKING OF CONVOLUTIONAL NEURAL NETWORKS FOR FACIAL EXPRESSIONS RECOGNITION," *J Theor Appl Inf Technol*, vol. 98, no. 18, 2020, [Online]. Available: www.jatit.org
[10] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Process Mag*, vol. 29, no. 6, pp. 141–142, 2012, doi: 10.1109/MSP.2012.2211477.
[11] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.
[12] TensorFlow Developers, "Keras Applications - TensorFlow Core," TensorFlow Documentation. Accessed: Aug. 08, 2025. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf
[13] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," Mar. 2022.
[14] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," Jun. 2021.
[15] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017.
[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 2019.
[17] A. Howard *et al.*, "Searching for MobileNetV3," Nov. 2019.
[18] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," Apr. 2018.
[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Dec. 2015.
[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Aug. 2016.
[21] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.